electronics

MDPI

Article

Lightweight Failover Authentication Mechanism for IoT-Based Fog Computing Environment

Soumya Banerjee [1], Ashok Kumar Das [2], Samiran Chattopadhyay [1], Sajjad Shaukat Jamal [3], Joel J. P. C. Rodrigues [4,5] and Youngho Park [6,*,†]

[1] Department of Information Technology, Jadavpur University, Salt Lake City, Kolkata 700 098, India; soumyabanerjee@outlook.in (S.B.); samiranc@gmail.com (S.C.)
[2] Center for Security, Theory and Algorithmic Research, International Institute of Information Technology, Hyderabad 500 032, India; ashok.das@iiit.ac.in
[3] Department of Mathematics, College of Science, King Khalid University, Abha 61413, Saudi Arabia; shussain@kku.edu.sa
[4] Federal University of Piauí (UFPI), Teresina-Pi 64049-550, Brazil; joeljr@ieee.org
[5] Instituto de Telecomunicações, 6201-001 Covilhã, Portugal
[6] School of Electronics Engineering, Kyungpook National University, Daegu 41566, Korea
* Correspondence: parkyh@knu.ac.kr; Tel.: +82-53-950-5114
† Current address: School of Electronics Engineering, Kyungpook National University, 80 Daehak-ro, Sangyeok-dong, Buk-gu, Daegu 41566, Korea.

Abstract: Fog computing as an extension to the cloud computing infrastructure has been invaluable in enhancing the applicability of the Internet of Things (IoT) paradigm. IoT based Fog systems magnify the range and minimize the latency of IoT applications. However, as fog nodes are considered transient and they offer authenticated services, when an IoT end device loses connectivity with a fog node, it must authenticate freshly with a secondary fog node. In this work, we present a new security mechanism to leverage the initial authentication to perform fast lightweight secondary authentication to ensure smooth failover among fog nodes. The proposed scheme is secure in the presence of a current *de-facto* Canetti and Krawczyk (CK)-adversary. We demonstrate the security of the proposed scheme with a detailed security analysis using formal security under the broadly recognized Real-Or-Random (ROR) model, informal security analysis as well as through formal security verification using the broadly-used Automated Validation of Internet Security Protocols and Applications (AVISPA) software tool. A testbed experiment for measuring computational time for different cryptographic primitives using the Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL) has been done. Finally, through comparative analysis with other related schemes, we show how the presented approach is uniquely advantageous over other schemes.

Keywords: Internet of Things (IoT); fog computing; failover; security; privacy; authentication

check for updates

Citation: Banerjee, S.; Das, A.K.; Chattopadhyay, S.; Jamal, S.S.; Rodrigues, J.J.P.C.; Park, Y. Lightweight Failover Authentication Mechanism for IoT-Based Fog Computing Environment. *Electronics* **2021**, *10*, 1417. https://doi.org/10.3390/electronics10121417

Academic Editor: George Angelos Papadopoulos

Received: 14 May 2021
Accepted: 9 June 2021
Published: 12 June 2021

1. Introduction

The adoption of Internet of Things (IoT) has been unprecedented. The concept has materialized into one of the most popular driver technology into the next generation of ubiquitous connectivity. As more applications of smart connectivity are realized even more applications are envisioned. The IoT paradigm aims to provide connectivity between the physical and the cyber worlds with the intention to enable greater economic welfare, accuracy, and efficiency with minimal human intervention [1,2]. IoT enables Industry 4.0, connectivity during a humanitarian crisis, and ushers in a more comfortable standard of living.

The IoT paradigm predicts an explosion of connected devices [3,4]. This, in spite of the distributed nature of IoT, put an unprecedented load on the existing centralized infrastructure. This issue is addressed with the fog computing paradigm and extension to cloud computing. Transient fog nodes can extend the connectivity of cloud computing infrastructure as well as reduce latency and pre-process data to reduce computational load [5]. Fog nodes, by design, form an intermediate layer between the cloud infrastructure and the IoT end devices. Figure 1 shows a fog architecture for edge-based IoT environment adapted form [6]. Fog nodes, by virtue of their deployment near the smart devices, offer location awareness, lower latency, capability for real-time interaction, and so forth. For example, in a smart vehicular network, fog nodes might be deployed regularly along the roadway. As vehicles pass by, they can communicate with the nearest fog node at that point. Thus, fog computing can provide added functionality to IoT systems.

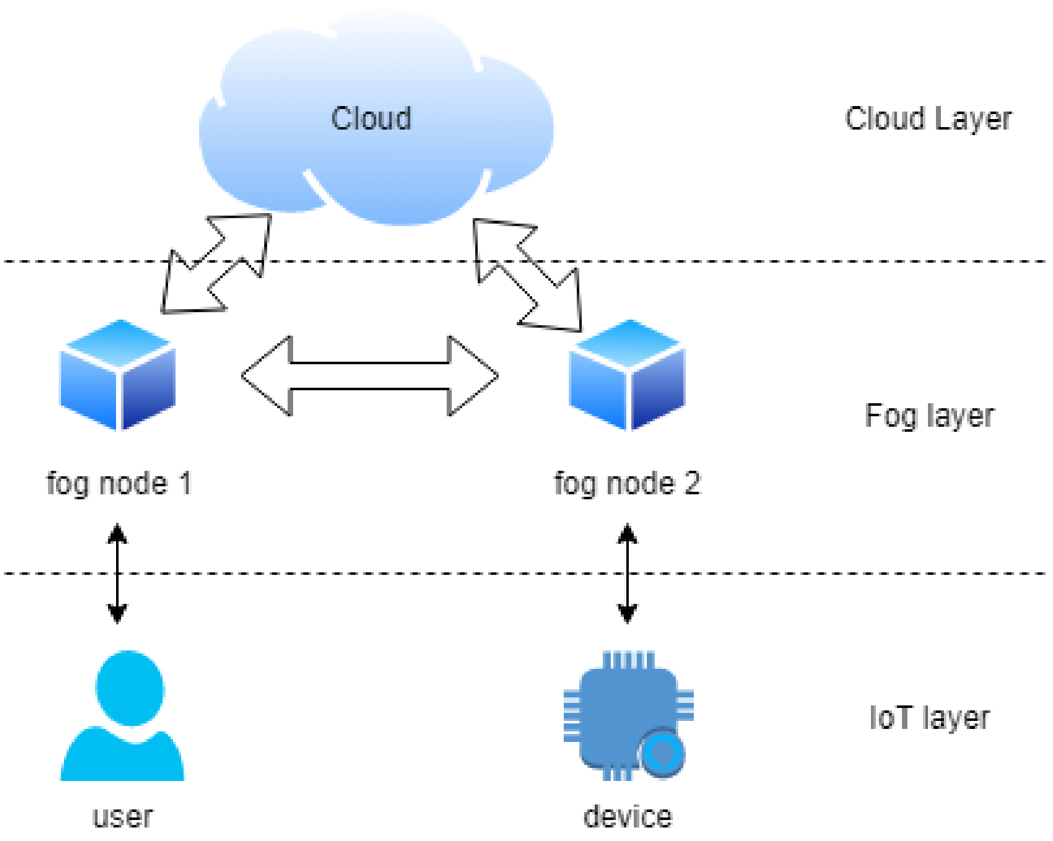


Figure 1. A generic fog architecture for edge-based IoT environment.

1.1. Motivation

Fog nodes, as described in Figure 1, act as a bridge between the end device and the cloud infrastructure. The fog nodes are considered transient and consequently, they are not trusted. The end device (for example, a user or an IoT smart device) must be authenticated by the fog node before any service is provided. This authentication can involve the cloud server, but that introduces latency as well at overhead at the cloud server. Normally, authentication and session key establishment between an end device and the fog nodes rely on public-key cryptography. However, relatively speaking, this is rather computationally expensive. Fog nodes are considered transient in nature. This can be because the nodes go out of range or they go off-line for some other reasons. In such a scenario, the end-user must re-authenticate with a secondary fog node that will take over the role of the previous fog node. The main objective with this work is to achieve the failover re-authentication without using public-key cryptography. This can be achieved if the fog nodes pre-agree on some security tokens, that are made available to the end device after the initial authentication. These security tokens can be leveraged to make a fast authentication possible between the end device and the secondary fog node. Moreover, insecure communication among the end devices and the fog nodes may lead to open up with several security attacks like replay, impersonation, man-in-the-middle and denial of service (clogging) attacks, that can be launched by an adversary. In order to resist such attacks, the designed failover authentication mechanism in this paper has been proposed. The proposed scheme also maintain anonymity and untraceability properties.

1.2. Research Contributions

The primary contributions in this work are summarized as follows:

- We present the architecture and threat model for the envisioned failover authentication model.
- We define the proposed failover authentication model in detail with all its phases.
- We present a detailed security analysis with both formal and informal security analysis, and also a formal security verification using the AVISPA software validation tool to prove the robustness of the proposed scheme against various known attacks.
- A testbed experiment for measuring computational time for different cryptographic primitives using the MIRACL library has been provided.
- Finally, through a detailed comparative study, we showcase the unique advantages of the proposed scheme.

1.3. Paper Outline

The rest of the paper is sketched as follows. The relevant related work is provided in Section 2. In Section 3, the architecture and threat models are discussed for analyzing the proposed scheme. Various phases related to the proposed scheme are then discussed in Section 4. A detailed security analysis is provided in Section 5 including the formal security verification through AVISPA simulation in Section 6 to show the robustness of the proposed scheme. A testbed experiment for measuring computational time for different cryptographic primitives using the MIRACL library is demonstrated in Section 7. After comparative among the proposed scheme and other existing relevant schemes in Section 8, the paper is finally concluded in Section 9.

2. Related Work

Access control and authentication are two important security services to secure different networking environments, like IoT, "Internet of Drones (IoD)", "Internet of Vehicles (IoV)", "Wireless Sensor Networks (WSNs)", cyber-physical systems, smart grids, healthcare services, and so forth [7–29]. Several existing works describe the procedure to secure establish an authenticated session key between the end devices, the fog nodes, and the cloud servers.

Wazid et al. [6] designed a "secure key management and user authentication scheme for fog computing environment, known as SAKA-FC". SAKA-FC establishes a common session key between a user, the fog node, and a smart device. The fog nodes could also establish a secure connection with the cloud. However, in their scheme, the cloud may act as a single point of failure. Though SAKA-FC is lightweight and offers several security and functionality features, it does not support failover authentication mechanism.

Roy et al. [9] proposed a user authentication for mobile cloud computing. It uses cryptographic hash, bitwise XOR, and fuzzy extractor as primitives. Though this scheme is secure and lightweight, it does not offer failover authentication mechanism. Similarly, there are other authentication schemes, such as the schemes proposed in [14,19,22–24,28], which are efficient and also secure, but they do not offer failover authentication feature.

Gope [30] presented a scheme for an anonymous device to device (D2D) authentication for fog computing environment. In Gope's scheme [30], there are three scenarios: (1) **LAAP1:** It is for the "initial authentication protocol for device-to-device (D2D)-aided fog computing"; (2) **LAAP2:** It denotes the "subsequent authentication protocol with the co-operation of EDs in D2D-aided fog computing"; and (3) **LAAP3:** It corresponds to the "subsequent authentication protocol with the co-operation of NADs in D2D-aided fog computing", where ED is an "end device"; NAD means the "Network Access Devices" and CCS denotes "Centralized Cloud Servers". The initial authentication involves the end devices, fog nodes as well as the cloud server. Subsequently, D2D authentication could be operated with the involvement of the cloud server. If the fog node becomes unavailable, the initial authentication must be repeated with a new fog node. Thus, this scheme doe not provide the fog failover authentication process. In addition, this scheme does not protect the "Ephemeral Secret Leakage (ESL)" attack under the "Canetti and Krawczyk (CK)-adversary model" [31].

Concone et al. [32] presented a cloud-sensing scheme, called "secure protocol for mobile crowdsensing (SMCP)", for fog based applications that utilized signatures and did not involve the cloud server until the final update. SMCP is based on ECC, "extended triple Diffie–Hellman key agreement" and "symmetric cryptography". However, in their scheme, the ESL attack under the CK-adversary model is not addressed.

Basudan et al. [33] suggested an improved "certificateless aggregate signcryption scheme (CLASC)" approach for a "privacy-preserving vehicular crowdsensing-based road surface condition monitoring system" that applies bilinear pairing operations. Since the signcryption requires time-consuming pairing operations, Cui et al. [34] presented a scheme for road monitoring based on fog computing to reduce the computational complexity.

Guo et al. [35] then proposed a fog-centric authenticated key agreement scheme without involving the trusted parties. Their scheme is very attractive as it does not need the involvement of a trusted cloud server. Unfortunately, their scheme was designed with the DY-adversary model, and it is secure against the ESL attack under the CK-adversary model.

Ali et al. [36] presented a secure authentication scheme for fog computing specifically resistant against clogging attacks. However, the authentication process in their scheme requires the involvement of the cloud server.

3. System Models

This section details the network model envisioned for the proposed system as well as the threat model describing an adversary's capabilities that the scheme is designed to be resilient against various known attacks.

3.1. Network Model

The fog architecture envisioned for this work, as shown in Figure 1, has been adapted from [6]. The cloud infrastructure is pre-deployed and is considered to be semi-trusted. The fog nodes are deployed by a fully-trusted registration authority, say RA, which is a component of the trusted cloud infrastructure. Thus, the fog nodes can securely communicate with each other leveraging their trusts within the RA. The pre-processed data can be forwarded by the fog nodes to the cloud server(s). The end devices, which are the users' device or IoT endpoints, communicate with the fog nodes acting as the gateway nodes. The need for expensive direct communication between end device and cloud infrastructure can be avoided. This is specially usefully when two end devices need to establish secure communication. This architecture also precludes the need for key management mechanism at the cloud infrastructure as the services offered are available only after post-authentication through the fog nodes.

In this work, we focus only on the fog and IoT layers. We aim to avoid the involvement of the cloud server in the authentication to minimize the communication and computational overheads. Similarly, as described in the motivation section, we also aim to reduce the overheads for the end devices.

3.2. Threat Model

The *de-facto* standard, known as the Dolev-Yao (DY) threat model [37] considers that an adversary, being a passive or an active adversary, has complete control over the communication media. Any message sent over the open channel is considered insecure. The adversary can eavesdrop on all messages transmitted. Additionally, the adversary can block, replay, or even modify any token transmitted over the channel. For this work, we adhere to another more stringent adversary model, known as the "Canetti and Krawczyk

(CK)-adversary model" [31]. The CK-adversary not only has all capabilities of the DY-adversary, but he/she can subvert secure information like ephemeral session states and secret keys through session hijacking attacks. Some end devices may be stolen or physically captured by the adversary, and he/she can learn the stored credentials from those devices through the differential power analysis attacks [38] and utilize the extracted credentials later for subsequent attacks on the system. The registration authority (RA) is a fully trusted entity, but the fog nodes are considered semi-trusted entities in the network.

4. Proposed Failover Authentication Scheme

This section details the proposed failover authentication scheme for an IoT-based fog computing environment.

The core idea behind the failover authentication is that if and when a fog node becomes unavailable, the end devices connected to it should be trivially able to switch over to a secondary fog node. The functionality provided by this scheme is a fast re-authentication with the secondary fog node without the need to go through an expensive public-key based authentication. The scheme has three per-requisite phases and the authentication phases. Before authentication, the system must be set up and the fog nodes must be enrolled and the end devices must register with the registration authority (RA). The *inter-fog node pre-agreement* must also be completed before *fast authentication*. Table 1 summarizes the important notations that are used in the proposed scheme, and their significance.

Table 1. Important notations and their significance.

Symbol	Description
RA	A fully-trusted registration authority
$\mathcal{U}$	An end device
FN	A semi-trusted fog node
p	A sufficiently large prime number (i.e., 160-bit number)
$\mathbb{Z}_p$	A finite (prime) field, $\mathbb{Z}_p = \{0, 1, \cdots, p-1\}$
$\mathbb{E}_p$	A non-singular elliptic curve over a prime finite field $\mathbb{Z}_p$
Q_x, k_x, ID_x	Public and private key and identity of entity x, respectively
ZY, YZ	Pre-shared security token for fast authentication
$h(\cdot)$	Collision-resistant cryptographic one-way hash function
$\|\|, \oplus$	Concatenation and bitwise XOR operations, respectively
$\mathcal{A}$	A passive or an active adversary
$\Rightarrow$	A secure channel
$\rightarrow$	A public (insecure) channel

The detailed description of each phase is provided in the following subsections.

4.1. Setup Phase

During the *Setup*, the registration authority (RA) selects a non-singular elliptic curve $\mathbb{E}_p$ of the form: $y^2 = x^3 + \alpha x + \beta \pmod{p}$ over a prime finite field $\mathbb{Z}_p$, where p is a large prime and the condition for non-singularity $4\alpha^3 + 27\beta^2 \neq 0 \pmod{p}$ is fulfilled. Then, the RA selects a generator G of an order n over $\mathbb{E}_p$ such that $n \cdot G = \mathcal{O}$, the zero point or point at infinity, and $n \cdot G = G + G + \cdots G\,(n\,times)$ represents an "elliptic curve point (scalar) multiplication". The RA also selects its own "elliptic curve cryptography (ECC)-based private key" k_R and computes the corresponding public key $Q_R = k_R \cdot G$. Finally, the RA picks a "collision-resistant cryptographic one way hash function, $h(\cdot)$" (for example, a Secure Hash Algorithm (SHA-1) [39]) and makes $h(\cdot)$, G and Q_R as public.

4.2. Fog Node Enrollment Phase

After the *Setup* phase, the individual fog nodes may be enrolled with the registration authority (RA). A fog node, say $Fog\ \ node\ \ x$, selects its ECC-based private-public keys pair (k_{F_x}, Q_{F_x}), where $Q_{F_x} = k_{F_x} \cdot G$, and learns the list of public keys for the other fog nodes. During the enrollment, the RA also transmits a list containing the mapping between obscured end-devices identities EID_u and their public keys Q_u, for all registered end devices. The list is periodically updated to intimate the fog nodes of newly registered end devices and fog nodes.

4.3. End Device Registration

After *Setup*, the end device (user's access device or otherwise) is registered with the system.

- The end device provides its identity (user's identity or otherwise), ID_u and selects $k_U \in \mathbb{Z}_p$ as its private key. Then it computes its public key $Q_u = k_U \cdot Q_R$ and securely transmits $\langle ID_u, Q_u \rangle$ to the RA.
- On receiving the registration request, RA selects a $x \in \mathbb{Z}$ and computes the obscured end-device identity $EID_u = h(ID_u || x)$. RA securely transmits $\langle EID_u \rangle$ to the end device saves $\langle ID_u, EID_u, Q_u \rangle$.
- The credentials in the end device can be secured with multi-factor authentication, which is beyond the scope of the scheme.

Figure 2 summarizes the *end device registration*. Note that, a secure channel is used as a conceptual term. For end devices, this can mean pre-deployment configuration and for end devices this referees to in-person registrations. After the initial registration, a variant of the registration, that does not require a secure channel, can be repeated periodically to update EID_u and Q_u.

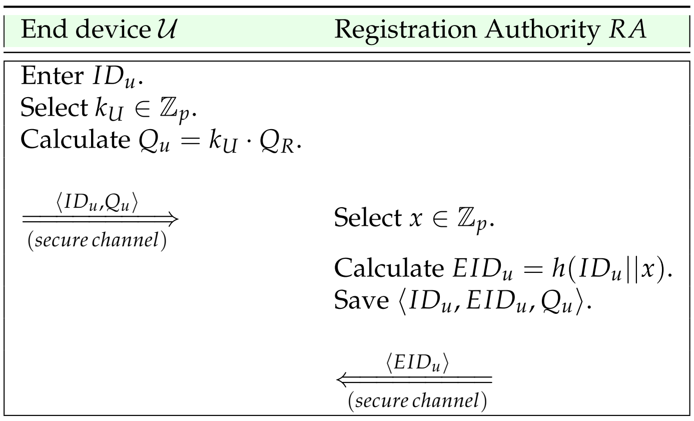

End device $\mathcal{U}$
Registration Authority RA
Enter ID_u.
Select $k_U \in \mathbb{Z}_p$.
Calculate $Q_u = k_U \cdot Q_R$.
$\xrightarrow[(secure\, channel)]{\langle ID_u, Q_u \rangle}$
Select $x \in \mathbb{Z}_p$.
Calculate $EID_u = h(ID_u || x)$.
Save $\langle ID_u, EID_u, Q_u \rangle$.
$\xleftarrow[(secure\, channel)]{\langle EID_u \rangle}$

Figure 2. Summary of end device registration.

4.4. Inter-Fog Node Pre-Agreement

Before the *fast authentication phase* is possible, all pairs of (adjacent) fog nodes must agree to own two security tokens, namely YZ and ZY. The pre-agreement is a cooperative procedure and can be initialized by either of the participating nodes. We describe this phase with the assumption that *Fog node 1* will act as an initiator and *Fog node 2* will act as a responder. The following are the steps that are executed:

- *Fog node 1* selects a random secret $y \in \mathbb{Z}_p$ and computes $Q_{12} = k_{F_1} \cdot Q_{F_2}$. Additionally, it calculates $K_y = Q_{12} \oplus y$, sets TS_1 as the current timestamp and then computes $V_1 = h(K_y || Q_{12} || TS_1)$ in order to transmit a message $M_{p_1} = \langle K_y, V_1, TS_1 \rangle$ to *Fog node 2* via open channel.
- On receiving the message M_{p_1}, *Fog node 2* computes $Q_{12} = k_{F_2} \cdot Q_{F_1}$ and verifies if V_1 is equal to $h(K_y || Q_{12} || TS_1)$. Only if the verification holds, it proceeds ahead. It then recovers $y = Q_{12} \oplus K_y$, selects a random secret $z \in \mathbb{Z}_p$ and computes $K_z = Q_{12} \oplus z$. It also sets TS_2 as the current timestamp, computes $V_2 = h(K_z || Q_{12} || TS_2)$ and transmits the message $M_{p_2} = \langle K_z, V_2, TS_2 \rangle$ to *Fog node 1* via an open channel. Moreover, it computes the security tokens $YZ = h(y || z)$ and $ZY = h(z || y)$.

- On receiving the message M_{p_2}, *Fog node 1* verifies if V_2 is equal to $h(K_z||Q_{12}||TS_2)$. If it is valid, it recovers $z = Q_{12} \oplus K_z$, and computes the security tokens $YZ = h(y||z)$ and $ZY = h(z||y)$.

Figure 3 summarizes the *inter-fog node pre-agreement* phase. This phase can be repeated periodically to update the security tokens as well.

Fog node 1 $< k_{F_1} >$	Fog node 2 $< k_{F_2} >$
Pick $y \in \mathbb{Z}_p$. Calculate $Q_{12} = k_{F_1} \cdot Q_{F_2}$, $K_y = Q_{12} \oplus y$, $V_1 = h(K_y||Q_{12}||TS_1)$.	
$\xrightarrow{M_{p_1} = \langle K_y, V_1, TS_1 \rangle}$ (open channel)	Compute $Q_{12} = k_{F_2} \cdot Q_{F_1}$. If $h(K_y||Q_{12}||TS_1) \neq V_1$ *terminate* Compute $y = Q_{12} \oplus K_y$. Select $z \in \mathbb{Z}_p$. Calculate $K_z = Q_{12} \oplus z$, $V_2 = h(K_z||Q_{12}||TS_2)$.
If $h(K_z||Q_{12}||TS_2) \neq V_2$ *terminate* Compute $z = Q_{12} \oplus K_z$. Save $YZ = h(y||z)$, $ZY = h(z||y)$.	$\xleftarrow{M_{p_2} = \langle K_z, V_2, TS_2 \rangle}$ (open channel) Save $YZ = h(y||z)$, $ZY = h(z||y)$.

Figure 3. Summary of Fog node pre-agreement.

4.5. Initial Authentication

An end device must first authenticate with any one fog node by relying on the public key cryptography. In this case, we have applied ECC-based public key infrastructure. This section describes the initial authentication between an end device, say $\mathcal{U}$ and a fog node, say *Fog node 1*. The following are the executed steps:

- The end device $\mathcal{U}$ selects a random secret $a \in \mathbb{Z}_p$ and computes $Q_{uF_1} = a \cdot Q_{F_1}$, $Q_a = a \cdot Q_R$ and $QID = Q_{uF_1} \oplus EID_u$. Additionally, it sets TS_1 as the current timestamp, computes $V_1 = h(QID||Q_{uF_1}||TS_1)$ and transmits a message $M_{a_1} = \langle QID, Q_a, V_1, TS_1 \rangle$ to *Fog node 1* over an insecure channel.
- On receiving the message M_{a_1}, the *Fog node 1* computes $Q_{uF_1} = k_{F_1} \cdot Q_a$ and verifies if V_1 is equal to $h(QID||Q_{uF_1}||TS_1)$. Only if the check holds, it proceeds ahead. It then recovers $EID_u = Q_{uF_1} \oplus QID$, looks up Q_u from EID_u, selects another random secret $b \in \mathbb{Z}_p$, and also computes $Q_b = b \cdot k_{F_1} \cdot Q_u$, $B = b \oplus h(EID_u||Q_{uF_1})$ and $SK = h(Q_{uF_1}||Q_b)$. Moreover, it sets TS_2 as the current timestamp, computes $V_2 = h(b||Q_u||SK||TS_2)$ and transmits the message $M_{a_2} = \langle B, V_2, TS_2 \rangle$ back to $\mathcal{U}$ via an open channel.
- On receiving the message M_{a_2}, $\mathcal{U}$ computes $b = B \oplus h(EID_u||Q_{uF_1})$, $Q_b = b \cdot k_U \cdot Q_{F_1}$ and $SK' = h(Q_{uF_1}||Q_b)$. If V_2 is equal to $h(b||Q_u||SK'||TS_2)$, $\mathcal{U}$ sets $SK = SK'$

Figure 4 summarizes the *initial authentication* phase. After this authentication, *Fog node 1* shares the security tokens for the nearby fog nodes that can operate as an failover. As these security tokens are transient, they must be periodically updated.

End device $\mathcal{U} < k_U >$	Fog node 1 $< k_{F_1} >$
Select $a \in \mathbb{Z}_p$. Compute $Q_{uF_1} = a \cdot Q_{F_1}$, $Q_a = a \cdot Q_R$, $QID = Q_{uF_1} \oplus EID_u$, $V_1 = h(QID\|\|Q_{uF_1}\|\|TS_1)$.	
$\xrightarrow{M_{a_1} = \langle QID, Q_a, V_1, TS_1 \rangle}$ (open channel)	Calculate $Q_{uF_1} = k_{F_1} \cdot Q_a$. If $h(QID\|\|Q_{uF_1}\|\|TS_1) \neq V_1$ *terminate* Compute $EID_u = Q_{uF_1} \oplus QID$, $Q_u = lookup(EID_u)$. Select $b \in \mathbb{Z}_p$. Compute $Q_b = b \cdot k_{F_1} \cdot Q_u$, $B = b \oplus h(EID_u\|\|Q_{uF_1})$, $SK = h(Q_{uF_1}\|\|Q_b)$, $V_2 = h(b\|\|Q_u\|\|SK\|\|TS_2)$.
	$\xleftarrow{M_{a_2} = \langle B, V_2, TS_2 \rangle}$ (open channel)
Calculate $b = B \oplus h(EID_u\|\|Q_{uF_1})$, $Q_b = b \cdot k_U \cdot Q_{F_1}$, $SK' = h(Q_{uF_1}\|\|Q_b)$. If $h(b\|\|Q_u\|\|SK'\|\|TS_2) = V_2$ $SK = SK'$	

Figure 4. Summary of initial authentication.

4.6. Fast Authentication

Once an end device $\mathcal{U}$ has learned the security token for the *Fog node 2*, if, for some reasons, the *Fog node 1* becomes unavailable or it goes out of communication range, $\mathcal{U}$ can leverage the security token to authenticate with *Fog node 2* without relying on the public key cryptography (as described in Section 4.5). This section describes the fast authentication between the end device $\mathcal{U}$ and the *Fog node 2* with the help of the following steps:

- The end device $\mathcal{U}$ selects a random secret $a \in \mathbb{Z}_p$ and computes $Q_a = a \oplus h(YZ||TS_1)$, $Q_{uF_2} = h(a||ZY)$, $Q_a = a \cdot h(YZ||TS_1)$ and $QID = Q_{uF_2} \oplus EID_u$, where TS_1 is the current timestamp. It then computes $V_1 = h(QID||Q_{uF_2}||TS_1)$ and transmits a message $M_{f_1} = \langle QID, Q_a, V_1, TS_1 \rangle$ to *Fog node 1* via a public channel.
- On receiving the message M_{a_1}, the *Fog node 2* computes $a = Q_a \oplus h(YZ||TS_1)$, recovers $Q_{uF_2} = h(a||ZY)$ and verifies if V_1 is equal to $h(QID||Q_{uF_2}||TS_1)$. Only if the check holds, it proceeds ahead. It recovers $EID_u = Q_{uF_2} \oplus QID$, looks up Q_u from EID_u, selects a random secret $b \in \mathbb{Z}_p$ and computes $Q_b = h(b||ZY||Q_u)$, $B = b \oplus h(EID_u||Q_{uF_2})$ and $SK = h(Q_{uF_2}||Q_b)$, where TS_2 is the current timestamp. Moreover, it computes $V_2 = h(b||Q_u||SK||TS_2)$ and transmits the message $M_{f_2} = \langle B, V_2, TS_2 \rangle$ back to $\mathcal{U}$ via a public channel.
- On receiving the message M_{f_2}, $\mathcal{U}$ recovers $b = B \oplus h(EID_u||Q_{uF_2})$, and computes $Q_b = h(b||ZY||Q_u)$ and $SK' = h(Q_{uF_2}||Q_b)$. If V_2 is equal to $h(b||Q_u||SK||TS_2)$, $\mathcal{U}$ sets the session key shared with the *Fog node 2* ass $SK = SK'$.

Figure 5 summarizes the *fast authentication* phase. After this authentication, the *Fog node 2* must also periodically share the security tokens for the nearby fog nodes that can operate as its failover.

End device $\mathcal{U} < k_U, ZY, YZ >$	Fog node 2 $< ZY, YZ >$
Generate $a \in \mathbb{Z}_p$.	
Calculate $Q_{uF_2} = h(a\|\|ZY)$,	
$Q_a = a \oplus h(YZ\|\|TS_1)$,	
$QID = Q_{uF_2} \oplus EID_u$,	
$V_1 = h(QID\|\|Q_{uF_2}\|\|TS_1)$.	
$\xrightarrow{M_{f_1} = \langle QID, Q_a, V_1, TS_1 \rangle}$	Generate $a = Q_a \oplus h(YZ\|\|TS_1)$.
(open channel)	Calculate $Q_{uF_2} = h(a\|\|ZY)$.
	If $h(QID\|\|Q_{uF_2}\|\|TS_1) \neq V_1$
	$terminate$
	Compute $EID_u = Q_{uF_1} \oplus QID$,
	$Q_u = lookup(EID_u)$.
	Generate $b \in \mathbb{Z}_p$.
	Calculate $Q_b = h(b\|\|ZY\|\|Q_u)$,
	$B = b \oplus h(EID_u\|\|Q_{uF_2})$,
	$SK = h(Q_{uF_2}\|\|Q_b)$,
	$V_2 = h(b\|\|Q_u\|\|SK\|\|TS_2)$.
	$\xleftarrow{M_{f_2} = \langle B, V_2, TS_2 \rangle}$
Calculate $b = B \oplus h(EID_u\|\|Q_{uF_2})$,	(open channel)
$Q_b = h(b\|\|ZY\|\|Q_u)$,	
$SK' = h(Q_{uF_2}\|\|Q_b)$.	
If $h(b\|\|Q_u\|\|SK\|\|TS_2) = V_2$	
$SK = SK'$	

Figure 5. Summary of fast authentication.

Remark 1. *After the enrollment in Section 4.2, the fog nodes must perform the mutual pre-agreement phase as described in Section 4.4 in order to support the fast authentication phase in Section 4.6. Similarity, an end device must be registered as described in Section 4.3, and it needs to perform the initial authentication and will be in an active session with a fog node. The end device may avail a fast authentication as described in Section 4.6 with an adjacent fog node based on necessity. Note that, the fast authentication is not possible if the device is not already in an active session because the fast authentication leverages the security of the initial authentication.*

5. Security Analysis

An authentication scheme can be susceptible to several vulnerabilities that enable an adversary to subvert the scheme. In this section, we first analyze the proposed scheme for formal security using the widely recognized random oracle model, known as the "Real-Or-Random (ROR) model" [40]. We then informally discuss how the proposed scheme resists various known attacks. Additionally, we report the simulation results under the automated software validation tool, called the "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [41] for the formal security verification and show that the proposed scheme is safe against passive/active attacks like "replay", "impersonation" and "man in the middle" attacks.

5.1. Formal Security Analysis Using ROR Model

In this section, we utilize the Real-Or-Random (ROR) model proposed in [40] to analyze the semantic security of the proposed scheme. The important elements for the ROR model are given below.

Participants: Let π_U^u, $\pi_{FN1}^{f_1}$ and $\pi_{FN2}^{f_2}$ denote the u^{th}, f_1^{th} and f_2^{th} instances corresponding to a user $\mathcal{U}$, the fog nodes FN_1 and FN_2, respectively [42,43]. These are also termed as the random oracles.

Partnering: The instances π_U^u and π_{FN}^f are considered to be partnered when the following conditions are satisfied simultaneously: (1) they share a communication session id *sid* and (2) partial transcript of all message exchanged between them are unique.

Freshness: π_U^u and π_{SD}^f are fresh if the session key SK established between $\mathcal{U}$ and FN has not been revealed to an adversary $\mathcal{A}$ with the help of the *Reveal* query discussed below.

Adversary: The adversary $\mathcal{A}$ is assumed to have complete influence over the communication media. Thus, $\mathcal{A}$ can eavesdrop and also alter, delete and forge messages at will during communication. Additionally, $\mathcal{A}$ has the access to the following queries:

- $Execute(\pi^u)$: By this query, which models an eavesdropping attack, $\mathcal{A}$ can intercept all the transmitted messages among $\mathcal{U}$, FN_1 and FN_2.
- $Send(\pi^f, m)$: This query models an as an active attack and enables $\mathcal{A}$ to send a message, say msg to its participating instance π^f, and also to receive a response in reply.
- $Test(\pi^u, \pi^f)$: This query utilizes the indistinguishability in the ROR model [40] to determine the semantic security of the session key SK established between $\mathcal{U}$ and FN. TO begin with, $\mathcal{A}$ performs an unbiased coin toss c. Its outcome decides the result of the *Test* query. If SK is fresh, π^u or π^f produces SK upon the satisfaction of the condition $c = 1$ or a random number for the fulfillment of the condition $c = 0$. Otherwise, it returns a null value.
- $Reveal(\pi^u)$: Through this query, $\mathcal{A}$ can learn the session key SK between $\mathcal{U}$ and FN_1 or between $\mathcal{U}$ and FN_2.

Definition 1 (Semantic security of session key)**.** *According to the ROR model, $\mathcal{A}$ must distinguish between an instance's actual session key and a random key. To this goal, $\mathcal{A}$ can repeat the $Test(\cdot)$ query to π^u or π^f, and save the results to bit c. $\mathcal{A}$ wins the game if $c = c'$, where c' is a randomly guessed bit. The advantage of $\mathcal{A}$ in breaking the semantic security of the proposed authenticated key agreement (AKE), say $\mathcal{P}$ in time t_p is defined as*

$$Adv_{\mathcal{P},\mathcal{A}}^{AKE}(t_p) = |2.Pr[SUCCESS] - 1|,$$

where $SUCCESS$ represents an event such that $\mathcal{A}$ wins the game, that is, $Pr[SUCCESS] = Pr[c' = c]$.

Random oracle: All communicating entities in the proposed scheme including $\mathcal{A}$ will have access to a "collision-resistant hash function, $h(\cdot)$" that is modeled as a "random oracle, say $\mathcal{HO}$".

Definition 2 (Collision-resistant one-way cryptographic hash function [44])**.** *Let h: $\{0,1\}^* \to \{0,1\}^{l_b}$ be a "collision-resistant one-way hash function" which is a deterministic function. It takes a variable-length input $x \in \{0,1\}^*$ and returns a fixed-length output, $y = h(x) \in \{0,1\}^{l_b}$ of l_b bits. Assume that the "advantage of an adversary $\mathcal{A}$ in finding a hash collision" in time t_p be denoted by $Adv_{\mathcal{A}}^{Hash}(t_p)$. Then,*

$$Adv_{\mathcal{A}}^{Hash}(t_p) \quad = \quad Pr[(ip_1, ip_2) \in_R \mathcal{A} : ip_1 \neq ip_2, h(ip_1) = h(ip_2)],$$

where the pair $(ip_1, ip_2) \in_R \mathcal{A}$ means that the input strings ip_1 and ip_2 are randomly picked by $\mathcal{A}$. We say "an (η, t)-adversary $\mathcal{A}$ attacking the collision resistance of $h(\cdot)$" means that the execution time taken by $\mathcal{A}$ is at most t and $Adv_{\mathcal{A}}^{Hash}(t_p) \leq \eta$.

Definition 3 (Elliptic curve decisional Diffie-Hellman problem (ECDDHP))**.** *Let $G \in \mathbb{E}_p$ be a point in an elliptic curve $\mathbb{E}_p$. Then, the ECDDHP states that with a given quadruple $(G, l_1 \cdot G, l_2 \cdot G, l_3 \cdot G)$ to decide if $l_3 = l_1 * l_2$ or a "uniform value", where l_1, l_2 and l_3 are the scalars chosen randomly from $Z_p^* = \{1, 2, \cdots, p-1\}$.*

Security proof: By utilizing the definition of the "collision-resistant hash function" (defined in Definition 2), "elliptic curve decisional Diffie-Hellman problem (ECDDHP)" (defined in Definition 3) and the above described ROR model, Theorem 1 provides the semantic security of the proposed scheme against the adversary $\mathcal{A}$ the derive the session key during the communication.

Theorem 1. *Let $\mathcal{A}$ be a polynomial time adversary running in time t_p against the proposed scheme $\mathcal{P}$ under the ROR model. If $Adv^{AKE}_{\mathcal{P},\mathcal{A}}(t_p)$ denotes $\mathcal{A}$'s advantage in breaking $\mathcal{P}$'s semantic security in time t_p in order to derive the session key between a legal registered end device $\mathcal{U}$ and an accessed fog node FN_1 or FN_2, then*

$$Adv^{AKE}_{\mathcal{P},\mathcal{A}}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv^{ECDDHP}_{\mathcal{A}}(t_p),$$

where q_h is the number of hash queries, $|Hash|$ defines the range space of $h(\cdot)$ and $Adv^{ECDDHP}_{\mathcal{A}}(t_p)$ is the advantage of an adversary $\mathcal{A}$ in solving ECDDHP in polynomial time t_p.

Proof. Our proof is analogous to the proofs that were presented in [45,46]. We define G_0–G_2 as the three sequential games in which an event $SUCCESS_i$ denotes that the adversary $\mathcal{A}$ can successfully guess the bit c corresponding to the game G_j, $j \in [0,2]$. The details of these games are provided below.

- **Game** G_0: This game models an actual (real) attack on the proposed scheme, $\mathcal{P}$ by $\mathcal{A}$. Thus, the bit c is guessed at the beginning of G_0. Therefore, the semantic security of the proposed scheme defined in Definition 1, it follows that

$$Adv^{AKE}_{\mathcal{P},\mathcal{A}}(t_p) = |2.Pr[SUCCESS_0] - 1|. \quad (1)$$

- **Game** G_1: This game models an eavesdropping attack, where $\mathcal{A}$ can query *Execute* oracle to intercept the messages $M_{a_1} = \langle QID, Q_a, V_1, TS_1 \rangle$, $M_{a_2} = \langle B, V_2, TS_2 \rangle$, $M_{f_1} = \langle QID, Q_a, V_1, TS_1 \rangle$, and $M_{f_2} = \langle B, V_2, TS_2 \rangle$ during the initial and fast authentication processes. Afterwards, $\mathcal{A}$ can also query *Test* oracle and determine if the result is the actual session key SK or just simply a random number. Note that in the proposed scheme, $SK = h(Q_{uF_1}||Q_b) = h(Q_{uF_1}||Q_b) = SK'$ is the session key established between an end device $\mathcal{U}$ and fog node FN_1 during the initial authentication, and also $SK = h(Q_{uF_2}||Q_b) = h(h(a||ZY)||h(b||ZY||Q_u)) = SK'$ is the established session key between a user $\mathcal{U}$ and a fog node FN_2 during the fast authentication. In both cases, to compute SK, $\mathcal{A}$ must know the short term secrets (a and b) as well as long term secrets (k_u, YZ and ZY) simultaneously. Thus, only the intended user $\mathcal{U}$ and fog nodes FN_1 and FN_2 can compute SK. Therefore, $\mathcal{A}$'s probability of wining the game G_1 is not increased form G_0 through an eavesdropping attack. Consequently, both the games G_0 and G_1 are indistinguishable, and we have the following result:

$$Pr[SUCCESS_1] = Pr[SUCCESS_0]. \quad (2)$$

- **Game** G_2: Under this game, the *Send* and hash $\mathcal{HO}$ queries are simulated. This game is modeled as an active attack, where $\mathcal{A}$ can attempt to fool a legitimate participant into accepting a modified message. $\mathcal{A}$ is permitted to make repeated queries to the random oracles to examine the presence of hash collisions. However, since all the messages M_{a_1}, M_{a_2}, M_{f_1} and M_{f_2} contain unique single use values, hash coalition does not occur (see Definition 2) when $\mathcal{A}$ queries the *Send* oracle with the help of $h(\cdot)$. Moreover, to derive the session key $SK = h(Q_{uF_1}||Q_b) = h(Q_{uF_1}||Q_b) = SK'$ is the session key established between an end device $\mathcal{U}$ and fog node FN_1 during the initial authentication, the adversary $\mathcal{A}$ needs to solve the computational ECDDHP defined in Definition 2. It is worth noticing that both the games G_1 and G_2 are "indistinguishable" except the *Send* and hash queries are simulated in G_2 along with solving ECDDHP. Thus, by using the birthday paradox results and the advantage of $\mathcal{A}$ in solving ECDDHP, we have,

$$|Pr[SUCCESS_2] - Pr[SUCCESS_1]| \leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \tag{3}$$

Finally, to win the game G_2, $\mathcal{A}$ needs to guess bit c' after querying the *Test* oracle. Thus, it is clear that

$$|Pr[SUCCESS_2] = \frac{1}{2}. \tag{4}$$

From Equations (1)–(4), we have

$$\begin{aligned} \frac{1}{2}Adv_{\mathcal{P},\mathcal{A}}^{AKE}(t_p) &= |Pr[SUCCESS_0] - \frac{1}{2}| \\ &= |Pr[SUCCESS_1] - \frac{1}{2}| \\ &= |Pr[SUCCESS_1] - |Pr[SUCCESS_2]| \\ &\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{ECDDHP}(t_p). \end{aligned} \tag{5}$$

By solving Equation (5), we obtain the required result:

$$Adv_{\mathcal{P},\mathcal{A}}^{AKE}(t_p) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{ECDDHP}(t_p).$$

□

5.2. *Informal Security Analysis*

In this section, through informal security analysis, we demonstrate the security features of the proposed scheme as well as its resilience against well-known attacks.

5.2.1. Mutual Authentication

In the proposed authentication scheme, the initial authentication is a standard public key-based authentication and the end device and the fog node authenticate each other with the public-private key pairs. For fast authentication, The participants are mutually authenticated with the pre-shared security tokens. Here for brevity, we have shown a single pair of secret tokens but for real implementations, these pair of secret tokens can be scaled up such that there is a one-to-one correspondence between a user and a token pair.

5.2.2. Anonymity and Untraceability

The messages exchanged during the initial or the subsequent fast authentications do not contain any plaintext identifiable values for the adversary to identify the participants with. Additionally, all the values are composed of nonce or timestamps, making tracing attacks infeasible. This the proposed scheme guarantees anonymity and untraceability.

5.2.3. Forward and Backward Secrecy

Assuming that the adversary can somehow learn the session key SK along with all its contributing secret values Q_{uF_2} and Q_b under the CK-adversary model. No past or future sessions are compromised as all these values are independent and distinct across sessions. This is true for both the initial or the subsequent fast authentications. Similarly, if the security tokens are leaked, no existing or future sessions are compromised.

5.2.4. Ephemeral Secret Leakage (ESL) Attack

The session key SK is composed form both long and short-term keys. Thus the adversary cannot derive session key SK unless both short and long-term secrets are exposed at once. Thus, the proposed scheme is resilient against the "ESL attack".

5.2.5. Impersonation Attacks

The initial authentication is designed around the public key cryptography and thus the public keys (issued through the trusted RA) prevent impersonation attacks. For fast authentication, the security token is responsible for mutual authentication and if compromised can lead to successful impersonation. However. the mitigation strategies mention in the context of mutual authentication prevents such attacks.

5.2.6. Clogging Attacks

The fog nodes detect and terminate spurious authentication requests after a xor and one or two hash operations for initial and fast authentications receptively. Thus, denial of services through clogging attacks will be mostly ineffective against the proposed scheme.

Remark 2. *The scheme is designed to work in conjuncture with other systems to ensure security against stolen smart cards, privileged insiders, end device capture, and other similar attacks and are beyond the scope of this scheme. Thus, within its scope, the proposed scheme resists all known attacks.*

6. Formal Security Verification through AVISPA Simulation

In this section, we validate the security of the proposed scheme with the help of one of the most widely recognized automated software verification tools, known as the "Automated Validation of Internet Security Protocols and Applications (AVISPA)" [41]. AVISPA is a push-button tool for automatic verification of security protocols. It is widely accepted as the formal verification of a security scheme against the man-in-the-middle and replay attacks (also, indirectly impersonation attacks), and internally implements the Dolev-Yao (DY) threat model [37]. Thus, the adversary has not only capability to intercept the communicating messages, but also can delete, modify or insert fake messages during the communication.

To verify a security scheme with AVISPA, it must be specified in the *High Level Protocol Specification Language* (HLPSL) [47]. AVISPA distribution includes an inbuilt converter for conversion from HLPSL to IF, known as *Intermediate Format* (IF), for backends for evaluations. There are four backends in AVISPA, namely (1) On-the-fly Model-Checker (OFMC), which is responsible for "performing several symbolic techniques to explore the state space in a demand-driven way", (2) Constraint Logic based Attack Searcher (CL-AtSe), which provides "a translation from any security protocol specification written as transition relation in intermediate format into a set of constraints which are effectively used to find whether there are attacks on protocols", (3) SAT-based Model-Checker (SATMC), which builds "a propositional formula and then the formula is fed to a state-of-the-art SAT solver to verify whether there is an attack or not" and (4) Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP), which approximates the "intruder knowledge by using regular tree languages". Of these four backends, OFMC and CL-AtSe are widely used because they support implementation of various functions including bitwise XOR operations, whereas SATMC and TA4SP do not support the implementation of bitwise XOR operations. Finally, the IF is evaluated by these backends and the result is presented in the *Output Format* (OF). For more details regarding AVISPA and HLPSL [41] can be consulted.

6.1. Specifying the Roles

HLPSL is a role-based language and individual roles for end device, registration authority, and two fog nodes, primary and failover, are defined in addition to the compulsory session, environment, and goal roles. The basic roles for an end device, two fog nodes FN_1 and FN_2, and the gateway node RA are defined in Figures 6–9, respectively. The compulsory roles for the session and goal and environment are defined in Figure 10. In the HLPSL implementation, secret credentials are kept secret by the *secret* declaration. The witness and request (authentication) are done by the *witness* and *request* declarations, respectively. The privacy and authentication goals are achieved through *secrecy_of* and *authentication_on* statements, respectively.

```
%%% Role specification for the mobile user ED %%%
role endDevice ( ED, FN1, FN2, RA : agent,
                H: hash_func, SecureChannel: symmetric_key,
                SND, RCV: channel(dy))
 played_by ED
 def=
 local
  State: nat, IDu, EIDu, Qu, Ku, X: text,
  A, Qf1, Qf2, Quf1, Qa, QID, Va1, TSa1, B,
  Bb, Qb, SK, Va2, TSa2, YZ, ZY: text,
  Af, TSf1, Quf2,  Vf1, Bf, Bbf, TSf2,  Vf2: text
 init State := 0
 transition
 % End device Registration phase
 1.State = 0  /\ RCV(start) =|>
  State' := 10/\ IDu':= new()
    /\ Ku' := new() /\ Qu' := exp(qr, Ku)
    /\ SND({IDu'.Qu'}_SecureChannel)
    /\ secret(IDu', sIDu, {ED,RA})
    /\ secret(Ku', sKu, {ED})
    /\ secret(YZ, sYZ, {FN1, FN2, ED})
    /\ secret(ZY, sZY, {FN1, FN2, ED})

 2.State = 10  /\ RCV( { H(IDu.X).Qf1.Qf2.H}_SecureChannel) =|>
  State' := 12 /\ secret(EIDu, sEIDu, {ED,RA,FN1,FN2})
 % End device initial authentication phase
    /\ A' := new() /\ TSa1':= new()
    /\ secret(A', sA, ED)
    /\ Quf1':= exp(Qf1, A) /\ Qa':= exp(qr, A)
    /\ QID':= xor(Quf1',EIDu) /\ Va1':= H(QID'.Quf1'.TSa1')
    /\ SND( QID'.Qa'.Va1'.TSa1')
    % ED has freshly generated the value Va1
      /\ witness (ED, FN1, sVa1, Va1')
    % ED has freshly generated the value Qa
    /\ witness (ED, FN1, sQa1, Qa')
 3.State = 12  /\ RCV(xor(B,H(EIDu'.Quf1')).H(B.Qu.SK'.TSa2').
                    TSa2'.{YZ.ZY}_SecureChannel ) =|>  %FIX THIS
  State':= 14 /\ B':= xor(Bb, H(EIDu.Quf1) )
    /\ Qb':= exp(exp(Quf1, Ku),B')
    /\ SK' := H(Quf1.Qb') /\ Va2':= H(B.Qu.SK'.TSa2)
    %  ED's calculated of the value Va2' matches value calculated by FN
     /\ request(ED, FN1, sVa2, Va2')
     %  ED's calculated of the value SK' matches value calculated by FN
     /\ request(ED, FN1, sSK1, SK')
  % End device Fast authentication phase
    /\ Af' := new() /\ TSf1':= new()
    /\ secret(Af', sAf, ED) /\ Quf2':= H(Af.ZY)
    /\ Qa' := xor(Af ,  H(YZ,TSf1'))
    /\ QID':= xor(Quf2',EIDu) /\ Vf1':= H(QID'.Quf2'.TSf2')
    /\ SND( QID'.Qa'.Va1'.TSf1')
    % ED has freshly generated the value Va1
      /\ witness (ED, FN2, sVf1, Vf1')
    % ED has freshly generated the value Qa
    /\ witness (ED, FN2, sQa2, Qa')
 4.State = 14  /\ RCV(xor(Bf,H(EIDu'.Quf2')).H(Bf.Qu.SK'.TSf2').TSf2) =|>
  State':= 15 /\ Bf':= xor( Bbf , H(EIDu.Quf1) )
    /\ Qb':= H(Bf'.ZY.Qu) /\ SK' := H(Quf2.Qb')
    /\ Vf2':= H(Bf.Qu.SK'.TSf2)
    %  ED's calculated of the value Va2' matches value calculated by FN1
     /\ request(ED, FN1, sVf2, Vf2')
     %  ED's calculated of the value SK' matches value calculated by FN1
     /\ request(ED, FN1, sSK2, SK')
 end role
```

Figure 6. HLPSL role specification for an end device.

In order to check the replay attack protection, it is required to mention the following two statements in the environment role:

```
session(ed, fn1, fn2, ra, h1, secureChannel)
/\ session(ed, fn1, fn2, ra, h1, secureChannel)
```

The man-in-the-middle and impersonation attacks are implemented by the following statements in the environment role, where an intruder (i) actively takes part in the communication:

```
session(ed, fn1, fn2, i,  h1, secureChannel)
/\ session(ed, fn1, i, ra, h1, secureChannel)
/\ session(ed, i, fn2, ra, h1, secureChannel)
/\ session(i, fn1, fn2, ra, h1, secureChannel)
```

```
%%% Role specification for the fog node 1 FN1 %%%
role fogNode1(ED, FN1, FN2, RA : agent, H: hash_func,
            SecureChannel: symmetric_key,
            SND, RCV   : channel(dy))
 played_by FN1
 def=
 local State : nat, Qu, IDu, X, Kf1, Qf1, Qf2: text,
  Y, TSp1, Q12, Vp1, Ky, Kz, TSp2, Vp2, Z, YZ, ZY: text,
  A, EIDu, QID, Quf1, TSa1, Va1, Qa, B, TSa2, Qb, Bb, SK, Va2 : text
 init State := 2
 transition
 % Fog node enrolement phase
 1.State =2  /\ RCV({Qu.H(IDu.X)}_SecureChannel) =|>
    % precomputation step
  State':=6 /\ Kf1':= new() /\ secret(Kf1', sKf1, {FN1})
    /\ secret(YZ', sYZ, {FN1, FN2, ED})
    /\ secret(ZY', sZY, {FN1, FN2, ED})
    /\ Qf1':=  exp(g, Kf1') /\ SND({Qf1'}_SecureChannel)
 2.State = 6  /\ RCV( {Qf2'}_SecureChannel ) =|>
 % Fog node precomputation phase
  State':= 8 /\ Y':= new() /\ TSp1' := new()
    /\ Q12':= exp(Qf2', Kf1) /\ Ky':= xor(Q12', Y')
    /\ Vp1':= H(Ky'.Q12', TSp1) /\ SND(Ky'.Vp1'.TSp1')
    % FN1 has freshly generated the value Vp1
    /\ witness (FN1, FN2, sVp1, Vp1')
    % FN1 has freshly generated the value Q12
    /\ witness (FN1, FN2, sQ12, Q12')
3.State = 8 /\ RCV( Kz'.H(Kz'.Q12', TSp2).TSp2' ) =|>
  State':= 11 /\ Vp2' := H(Kz'.Q12', TSp2) /\ Z':= xor(Q12, Kz')
    /\ YZ':= H(Y.Z') /\ ZY':= H(Z'.Y)
    %  FN1's received  value ERM matches the value calculated by FA2
     /\ request(FN1, FN2, sVp2, Vp2')
     %  FN1's calculated of the value YZ matches value calculated by FA2
     /\ request(FN1, FN2, ssYZ, YZ')
     %  FN1's calculated of the value ZY matches value calculated by FA2
     /\ request(FN1, FN2, ssZY, ZY')
     /\ SND({Qf1'}_SecureChannel) % control return to RA
 % End device initial authentication phase
 4.State = 11  /\ RCV(xor(exp(Qf1, A),EIDu).exp(qr, A).
                  H(QID'.Quf1'.TSa1').TSa1') =|>
  State':= 17 /\ Quf1':= exp(Qa, Kf1)
    /\ Va1':= H(QID'.Quf1'.TSa1')
    %  FN1's calculated of the value Qa matches value calculated by ED
     /\ request(ED, FN1, sQa1, Qa')
     %  FN1's calculated of the value Va1 matches value calculated by ED
     /\ request(ED, FN1, sVa1, Va1')
     /\ EIDu' := xor(Quf1',QID') /\ B' := new()
    /\ TSa2' := new() /\ secret(B', sB, FN1)
    /\ Qb' := exp(Qu, xor(B',Kf1)) /\ Bb':= xor(B,H(EIDu'.Quf1'))
    /\ SK' := H(Quf1'.Qb') /\ Va2':= H(B.Qu.SK'.TSa2')
    /\ SND( Bb'.Va2'.TSa2'.{YZ.ZY}_SecureChannel )
    % FN1 has freshly calculated the value SK'and  Va2'
    /\ witness (FN1, ED, sVa2, Va2') /\ witness (FN1, ED, sSK1, SK')
end role
```

Figure 7. HLPSL role specification for fog node 1 (FN_1).

```
%%% Role specification for the fog node 2 FN2 %%%
role fogNode2(ED, FN1, FN2, RA: agent, H: hash_func,
             SecureChannel  : symmetric_key,
             SND, RCV: channel(dy))
 played_by FN2
 def=
 local State : nat, Qu, IDu, X, Kf2, Qf1, Qf2: text,
  Y, TSp1, Q12, Vp1, Ky, Kz, TSp2, Vp2, Z, YZ, ZY: text,
  Af, EIDu, QID, Quf1, Quf2, TSf1, Va1, Qa, Bf,
  TSf2, Qb, Bbf, SK, Vf1, Vf2: text
 init State := 2
 transition
 % Fog node enrolement phase
 1.State =4  ∧ RCV({ Qu.H(IDu.X).Qf1}_SecureChannel) =|>
  State':=7 ∧ Kf2':= new() ∧ secret(Kf2', sKf2, {FN2})
    ∧ Qf2':=  exp(g, Kf2') ∧ SND({Qf2'}_SecureChannel)
 % Fog node precomputation phase
 2.State = 7 ∧ RCV( xor(Q12', Y').H(Ky'.Q12', TSp1).TSp1' ) =|>
  State':= 13 ∧ Q12':= exp(Qf1', Kf2) ∧ Vp1':= H(Ky'.Q12', TSp1)
%  FN2's received  value Vp1 matches the value calculated by FA1
     ∧ request(FN1, FN2, sVp1, Vp1')
%  FN1's calculated of the value Q12 matches value calculated by FA1
     ∧ request(FN1, FN2, sQ12, Q12')
     ∧ Y':= xor(Q12', Ky) ∧ Z':= new() ∧ TSp2' := new()
     ∧ Kz':= xor(Q12', Z') ∧ YZ':= H(Y'.Z')
     ∧ ZY':= H(Z'.Y') ∧ Vp2':= H(Kz'.Q12', TSp2)
    ∧ SND(Kz'.Vp2'.TSp2')
    % RA has freshly generated the value Ky
    ∧ witness (FN1, FN2, sVp2, Vp2')
    % RA has freshly generated the value Q12
    ∧ witness (FN1, FN2, ssYZ, YZ')
      % RA has freshly generated the value Q12
    ∧ witness (FN1, FN2, ssZY, ZY')
    ∧ secret(YZ', sYZ, {FN1, FN2, ED})
    ∧ secret(ZY', sZY, {FN1, FN2, ED})
 % End device fast authentication phase
 3.State = 13  ∧ RCV(xor(H(Af.ZY),EIDu).xor(Af,H(YZ,TSf1')).
                H(QID'.Quf2'.TSf1').TSf1') =|>
  State':= 18 ∧ Af' := xor(Qa,H(YZ,TSf1')) ∧ Quf2':= H(Af.ZY)
    ∧ Vf1':= H(QID'.Quf2'.TSf1')
%  FN2's calculated of the value Vfi   matches value calculated by ED
     ∧ request(ED, FN1, sQa2, Qa')
 %  FN2's calculated of the value Va1 matches value calculated by ED
     ∧ request(ED, FN1, sVa1, Va1')
     ∧ EIDu' := xor(Quf1',QID') ∧ Bf' := new() ∧ TSf2' := new()
     ∧ secret(Bf', sBf, FN2) ∧ Qb' := H(Bf.ZY.Qu)
    ∧ Bbf':= xor(Bf,H(EIDu'.Quf2')) ∧ SK' := H(Quf2'.Qb')
    ∧ Vf2':= H(Bf.Qu.SK'.TSf2') ∧ SND( Bbf'.Vf2'.TSf2')
    % FN2 has freshly calculated the value SK'and  Vf2'
    ∧ witness (FN2, ED, sVf2, Vf2') ∧ witness (FN2, ED, sSK2, SK')
end role
```

Figure 8. HLPSL role specification for fog node 2 (FN_2).

The simulation started with end device registration, described the fog node enrollments, the pre-arrangement of the security tokens, initial authentication, and the fast authentication with the failover fog node.

6.2. Simulation Results and Discussion

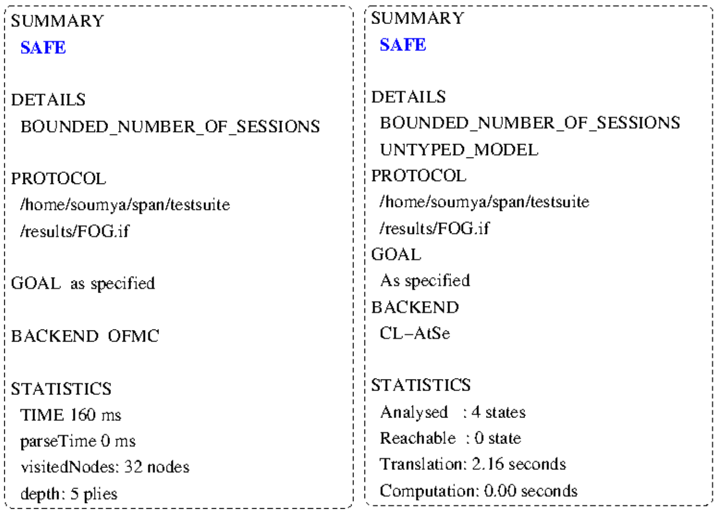
We evaluated the HLPSL script with the "SPAN, the *Security Protocol ANimator* for AVISPA" software tool [48]. Figure 11 presents the simulation results in *OF*. The results demonstrate the proposed scheme is secure against replay, man-in-the-middle and impersonation attacks.

```
%%% Role specification for the gateway node RA %%%
role redgAuth(ED, FN1, FN2, RA: agent, H: hash_func,
            SecureChannel: symmetric_key,
            SND, RCV: channel(dy))
 played_by RA
 def=
 local State: nat, IDu, EIDu, Ku, X, Qu, Qf1, Qf2: text
 init State := 1
 transition
 % User registration phase
 1.State = 1  /\ RCV({ IDu.exp(qr, Ku)}_SecureChannel) =|>
  State':= 3  /\ secret(IDu', sIDu, {ED,RA}) /\ X' := new()
    /\ EIDu':= H(IDu.X') /\ secret(X', sX, {RA})
 % Fog node 1 enrolement phase
    /\ SND({Qu.EIDu'}_SecureChannel)
 2.State = 3  /\ RCV({ Qf1 }_SecureChannel) =|>
 % Fog node 2 enrolement phase
  State':= 5 /\ SND({Qu.EIDu.Qf1'}_SecureChannel)
 3.State = 5  /\ RCV({ Qf2 }_SecureChannel) =|>
 % Fog node 1 enrolement phase, part2
  State':= 9 /\ SND( {Qf2'}_SecureChannel)
 % User registration phase
 4.State = 9 /\ RCV({ Qf1 }_SecureChannel) =|>
  State':= 16 /\ SND({ EIDu.Qf1.Qf2.H}_SecureChannel )
end role
```

Figure 9. HLPSL role specification for the gateway node (RA).

```
%%% Role specification for the session %%%
role session (ED, FN1, FN2, RA: agent,
    H: hash_func, SecureChannel: symmetric_key)
 def=
  local  S1, R1, S2, R2, S3, R3, S4, R4: channel (dy)
 composition
    endDevice(ED, FN1, FN2, RA, H, SecureChannel, S1, R1)
 /\  fogNode1(ED, FN1, FN2, RA, H, SecureChannel, S2, R2)
 /\  fogNode2(ED, FN1, FN2, RA, H, SecureChannel, S3, R3)
 /\  redgAuth(ED, FN1, FN2, RA, H, SecureChannel, S4, R4)
end role

%%% Role specification for the goal and environment %%%
role environment()
 def=
  const ed,fn1, fn2, ra: agent,
    secureChannel: symmetric_key, h1: hash_func, g, qr: text,
        sIDu, sKu, sEIDu, sA, sYZ, sZY,
        sAf, sKf1, sKf2, sB, sBf, sX,
        sSK1, sVf2, ssYZ, ssZY, sVp2, sVa2, sSK2, sVp1, sQ12,
        sVf1, sQa2, sQa1, sVa1 : protocol_id

      intruder_knowledge = {ed, fn1, fn2, ra, g, qr}

  composition
%%% Replay attack checking
    session(ed, fn1, fn2, ra, h1, secureChannel)
 /\ session(ed, fn1, fn2, ra, h1, secureChannel)
%%% Man−in−the−middle attack checking
 /\   session(ed, fn1, fn2, i,  h1, secureChannel)
 /\ session(ed, fn1, i, ra, h1, secureChannel)
 /\ session(ed, i, fn2, ra, h1, secureChannel)
 /\ session(i, fn1, fn2, ra, h1, secureChannel)
end role

goal
%%% Confidentiality or privacy
 secrecy_of sIDu, sKu, sEIDu, sA, sYZ, sZY,
          sAf, sKf1, sKf2, sB, sBf, sX
%%% Authentication
 authentication_on sSK1, sVf2, ssYZ, ssZY, sVp2, sVa2,
                sSK2, sVp1, sQ12, sVf1, sQa2, sQa1, sVa1
end goal
environment()
```

Figure 10. HLPSL role specification for the session, goal and environment.



```
SUMMARY                              SUMMARY
 SAFE                                 SAFE

DETAILS                              DETAILS
 BOUNDED_NUMBER_OF_SESSIONS           BOUNDED_NUMBER_OF_SESSIONS
                                      UNTYPED_MODEL
PROTOCOL                             PROTOCOL
 /home/soumya/span/testsuite          /home/soumya/span/testsuite
 /results/FOG.if                      /results/FOG.if
                                     GOAL
GOAL  as specified                    As specified
                                     BACKEND
BACKEND  OFMC                         CL-AtSe

STATISTICS                           STATISTICS
 TIME 160 ms                          Analysed   : 4 states
 parseTime 0 ms                       Reachable  : 0 state
 visitedNodes: 32 nodes               Translation: 2.16 seconds
 depth: 5 plies                       Computation: 0.00 seconds
```

Figure 11. Results of AVISPA simulation under OFMC and CL-AtSe backends.

7. Testbed Experiments Using MIRACL

In this section, we measure the execution time needed for different cryptographic primitives with the help of the broadly recognized "Multiprecision Integer and Rational Arithmetic Cryptographic Library (MIRACL)" [49].

The notations T_{senc}/T_{sdec}, T_{eca}, T_{ecm}, T_{fe}, T_h, T_{poly}, T_{mul} and T_{add} denote the time needed for computing symmetric encryption/decryption (using Advanced Encryption Standard (AES) algorithm [50]), elliptic curve point addition, elliptic curve point (scalar) multiplication, fuzzy extractor operation [51], one-way hash operation (using Secure Hash Algorithm (SHA-1) [39]), evaluation of an t-degree polynomial over a finite field, multiplication in a finite field and addition in a finite field, respectively. If we use the Horner's rule [52], the evaluation of an t-degree uni-variate polynomial, say $f(a, y)$ at $y = b$ (that is, $f(a, b)$) needs t modular multiplications and t modular additions, where a and b are taken from a finite field. In other words, $T_{poly} = t(T_{mul} + T_{add})$.

We have considered two platforms for the testbed experiment. In each platform, we execute each cryptographic primitive for 100 trials in order to measure the average run time of primitives.

- **Platform 1:** This platform corresponds to a server setting under the environment: "Ubuntu 18.04.4 LTS, with 7.7 GiB memory, Intel Core i7 processor- 8565U, CPU @ 1.80GHz $\times$ 8, 64-bit OS type and disk size 966.1 GB". The experimental results for different primitives are then measured using MIRACL library and then provided in Table 2. Note that under this platform, $T_{poly} = t(T_{mul} + T_{add}) = 0.006\,t$ milliseconds by considering average time. In addition, it is assumed that $T_{fe} \approx T_{ecm}$ [53].

Table 2. Execution time (in milliseconds) of cryptographic primitives under a server setting.

Primitive	Max. Time (ms)	Min. Time (ms)	Average Time (ms)
T_h	0.149	0.024	0.055
T_{senc}	0.008	0.002	0.003
T_{sdec}	0.005	0.002	0.003
T_{mul}	0.035	0.002	0.004
T_{add}	0.004	0.001	0.002
T_{ecm}	2.998	0.284	0.674
T_{eca}	0.002	0.001	0.002

- **Platform 2:** In this platform, we measured the execution time for cryptographic primitives by MIRACL for the smart device side under the Raspberry PI 3 setting. The system configuration is considered as follows: "Raspberry PI 3 B+ Rev 1.3, Ubuntu 20.04 LTS, 64- bit OS, 1.4 GHz Quad-core processor, cores 4, 1 GB RAM" [54]. In Table 3, we have tabulated the experimental results of different cryptographic primitives. Note that under this platform, $T_{poly} = t(T_{mul} + T_{add}) = 0.021t$ milliseconds by considering average time.

Table 3. Execution time (in milliseconds) of cryptographic primitives under Raspberry PI 3 setting.

Primitive	Max. Time (ms)	Min. Time (ms)	Average Time (ms)
T_h	0.643	0.274	0.309
T_{senc}	0.038	0.017	0.018
T_{sdec}	0.054	0.009	0.014
T_{mul}	0.016	0.009	0.011
T_{add}	0.013	0.008	0.010
T_{ecm}	4.532	2.206	2.288
T_{eca}	0.021	0.015	0.016

8. Comparative Study

In this section, we compare the proposed scheme with the recent authentication schemes proposed by Gope [30], Guo et al. [35] and Ali et al. [36]. For the proposed scheme, we have considered three cases as follows:

- **Case 1:** It denotes the "fog node pre-agreement" phase
- **Case 2:** It corresponds to the "initial authentication" phase
- **Case 3:** It represents the "fast authentication" phase

In Gope's scheme [30], we have the following three scenarios:

- **LAAP1:** It is for the "initial authentication protocol for device-to-device (D2D)-aided fog computing"
- **LAAP2:** It denotes the "subsequent authentication protocol with the co-operation of EDs in D2D-aided fog computing"
- **LAAP3:** It corresponds to the "subsequent authentication protocol with the co-operation of NADs in D2D-aided fog computing"

where ED is an "end device"; NAD means the "Network Access Devices" and CCS denotes "Centralized Cloud Servers".

8.1. Computation Costs Comparison

We have used the average execution time for different cryptographic primitives for the server setting and Raspberry PI 3 setting as shown in Tables 2 and 3 as those for an end device and a fog node/cloud server, respectively. In Table 4, we have compared the computational costs needed for the entities, like end device, fog node and cloud server during various authentication phases among the proposed scheme and other schemes, such as the schemes of Gope [30], Guo et al. and Ali et al. During the fast authentication process (Case 3), in the proposed scheme the end device and the fog nodes need to perform $7T_h \approx 2.163$ ms and $7T_h \approx 0.385$ ms, respectively. The cloud server is not involved in the proposed scheme. From the comparative analysis, it is observed that the proposed scheme has comparable computation overheads for various entities as compared to those for other related competing schemes.

Table 4. Computation costs comparison.

Scheme	Cost at ED	Cost at Fog Node (NAD)	Cost at Cloud Server (CCS)
Proposed (Case 1)	–	$2T_{ecm} + 8T_h$ $\approx$ 1.788 ms	–
Proposed (Case 2)	$3T_{ecm} + 4T_h$ $\approx$ 8.1 ms	$2T_{ecm}4T_h$ $\approx$ 1.568 ms	–
Proposed (Case 3)	$7T_h$ $\approx$ 2.163 ms	$7T_h$ $\approx$ 0.385 ms	–
Gope [30] (LAAP1)	$7T_h$ $\approx$ 2.163 ms	$3T_h$ $\approx$ 0.165 ms	$6T_h$ $\approx$ 0.33 ms
Gope [30] (LAAP2)	$11T_h$ $\approx$ 3.399 ms	$3T_h$ 0.165 ms	–
Gope [30] (LAAP3)	$5T_h$ $\approx$ 1.545 ms	$11T_h + 2T_{senc}/T_{sdec}$ 0.611 ms	–
Guo et al. [35]	$T_{fe} + 19T_h + 2T_{poly}$ $\approx (8.159 + 0.042t)$ ms	$8T_h + 5T_{poly}$ $\approx (0.44 + 0.03t)$ ms	–
Ali et al. [36]	$T_{fe} + 18T_h + 3T_{ecm}$ $\approx$ 14.714 ms	$4T_{ecm} + 8T_h + T_{eca}$ $\approx$ 3.138 ms	–

Note: ED: "end device"; NAD: "Network Access Devices"; CCS: "Centralized Cloud Servers"; LAAP1: "Initial authentication protocol for D2D-aided fog computing"; LAAP2: "Subsequent authentication protocol with the co-operation of EDs in D2D-aided fog computing"; LAAP3: "Subsequent authentication protocol with the co-operation of NADs in D2D-aided fog computing"; Case 1: "Fog node pre-agreement"; Case 2: "Initial authentication"; Case 3: "Fast authentication"; t: degree of a uni-variate polynomial over a finite field.

8.2. Communication Costs Comparison

For the communication overheads of different existing schemes, we assume that the hash digest, a random nonce (secret) and an identity to be 160 bits each. We additionally assume that a sequence number and a timestamp to be 32 bits long. In addition, AES-128 cipher [50] needs 128-bit plaintext/ciphertext block and an elliptic curve point needs $(160 + 160) = 320$ bits by assuming 160-bit ECC security remains the same level as that for 1024-bit RSA-based public key cryptosystem. In the proposed scheme, Case 1, Case 2 and Case 3 require 88 bytes, 148 bytes and 108 bytes, respectively, for exchange of 2 messages in each case. The communication overheads of the related schemes and the proposed scheme are compared in Table 5. We can clearly see that the proposed scheme has the lowest communication cost among the related existing schemes.

8.3. Security and Functionality Features Comparison

Table 6 summarizes the security and functionality features of the proposed scheme and the related schemes. We can see all the compared schemes support anonymity preserving mutual authentication and resist the known attacks. Apart from Ali et al.'s scheme [36], all other schemes support authentication without the cloud server. The schemes in [30] and [35] are not secure against ESL attack under the CK-adversary model [31]. Finally, as per the design motivation, only the proposed scheme supports fast failover authentication for fog nodes failure. Overall, the proposed scheme provides the richest security and functionality features while having comparable computational costs and lowest communication overheads, as compared to other competing schemes.

Table 5. Communication costs comparison.

Scheme	No. of Bytes	No. of Messages
Proposed (Case 1)	88	2
Proposed (Case 2)	148	2
Proposed (Case 3)	108	2
Gope [30] (LAAP1)	400	4
Gope [30] (LAAP2)	320	4
Gope [30] (LAAP3)	360	4
Guo et al. [35]	272	3
Ali et al. [36]	352	3

Note: LAAP1: "Initial authentication protocol for D2D-Aided fog computing"; LAAP2: "Subsequent authentication protocol with the co-operation of EDs in D2D-Aided fog computing"; LAAP3: "Subsequent authentication protocol with the co-operation of NADs in D2D-Aided fog computing"; Case 1: "Fog node pre-agreement"; Case 2: "Initial authentication"; Case 3: "Fast authentication".

Table 6. Security and functionality features comparison.

Functionality Features	Our	Gope [30]	Guo et al. [35]	Ali et al. [36]
Authentication without cloud servers	✓	✓	✓	✗
Fog failover authentication	✓	✗	✗	✗
Resists known attacks	✓	✓	✓	✓
Mutual authentication	✓	✓	✓	✓
ESL attack under CK-adversary model	✓	✗	✗	✓
Formal security verification	✓	✗	✗	✓

9. Concluding Remarks and Future Works

In this work, we have highlighted the need for a fast authentication mechanism in case of fog nodes failure. To achieve this goal, we have presented a new lightweight failover authentication mechanism for fog computing environment. We have shown the robustness of the proposed scheme with a detailed security analysis along with the formal security analysis under the ROR random oracle model, informal security analysis and also the formal security verification under the widely-accepted software validation tool, known as AVISPA. In addition, the testbed experiments for measuring computational time needed for various cryptographic primitives under the MIRACL library have been provided. Finally, through a comparative study among the proposed scheme and other related recent schemes, we have demonstrated the advantage of the proposed approach in terms of the security and functionality features, and communication and computational overheads. In future, we have planned to integrate the proposed scheme into a more complete fog enabled IoT architecture and evaluate the same in a real-world testbed scenario.

Author Contributions: Conceptualization, A.K.D., J.J.P.C.R. and Y.P.; Formal analysis, S.B.; Investigation, S.B., A.K.D., S.C. and S.S.J.; Methodology, S.B. and A.K.D.; Project administration, J.J.P.C.R. and Y.P.; Supervision, A.K.D., S.C., J.J.P.C.R. and Y.P.; Validation, S.C.; Visualization, S.S.J.; Writing—original draft, S.B.; Writing—review & editing, A.K.D. All authors have read and agreed to the published version of the manuscript.

Funding: The authors extend their gratitude to the Deanship of Scientific Research at King Khalid University for funding this work through research groups program under grant number R.G.P. 2/48/42. This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant 2020R1I1A3058605. This work was also partially funded by FCT/MCTES through national funds and when applicable co-funded EU funds under the Project UIDB/50008/2020; and by Brazilian National Council for Scientific and Technological Development - CNPq, via Grant No. 313036/2020-9.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: We thank the anonymous reviewers and the Editor for their valuable comments, which helped us to improve the quality and presentation of the paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Das, A.K.; Zeadally, S.; He, D. Taxonomy and analysis of security protocols for Internet of Things. *Future Gener. Comput. Syst.* **2018**, *89*, 110–125. [CrossRef]
2. Zeadally, S.; Das, A.K.; Sklavos, N. Cryptographic technologies and protocol standards for Internet of Things. *Internet Things* **2019**, *14*, 100075. [CrossRef]
3. Gubbi, J.; Buyya, R.; Marusic, S.; Palaniswami, M. Internet of Things (IoT): A vision, architectural elements, and future directions. *Future Gener. Comput. Syst.* **2013**, *29*, 1645–1660. [CrossRef]
4. Dizdarević, J.; Carpio, F.; Jukan, A.; Masip-Bruin, X. A Survey of Communication Protocols for Internet of Things and Related Challenges of Fog and Cloud Computing Integration. *ACM Comput. Surv.* **2019**, *51*, 1–29. [CrossRef]
5. Dastjerdi, A.V.; Gupta, H.; Calheiros, R.; Ghosh, S.; Buyya, R. Chapter 4—Fog Computing: Principles, architectures, and applications. In *Internet of Things*; Buyya, R., Vahid Dastjerdi, A., Eds.; Morgan Kaufmann: San Francisco, CA, USA, 2016; pp. 61–75.
6. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V. Design of secure key management and user authentication scheme for fog computing services. *Future Gener. Comput. Syst.* **2019**, *91*, 475–492. [CrossRef]
7. Wazid, M.; Das, A.K.; Bhat, K.V.; Vasilakos, A.V. LAM-CIoT: Lightweight authentication mechanism in cloud-based IoT environment. *J. Netw. Comput. Appl.* **2020**, *150*, 102496. [CrossRef]
8. Wazid, M.; Das, A.K.; Kumar, N.; Vasilakos, A.V.; Rodrigues, J.J.P.C. Design and Analysis of Secure Lightweight Remote User Authentication and Key Agreement Scheme in Internet of Drones Deployment. *IEEE Internet Things J.* **2019**, *6*, 3572–3584. [CrossRef]
9. Roy, S.; Chatterjee, S.; Das, A.K.; Chattopadhyay, S.; Kumar, N.; Vasilakos, A.V. On the Design of Provably Secure Lightweight Remote User Authentication Scheme for Mobile Cloud Computing Services. *IEEE Access* **2017**, *5*, 25808–25825. [CrossRef]
10. Jiang, Q.; Zeadally, S.; Ma, J.; He, D. Lightweight three-factor authentication and key agreement protocol for internet-integrated wireless sensor networks. *IEEE Access* **2017**, *5*, 3376–3392. [CrossRef]
11. Odelu, V.; Das, A.K.; Goswami, A. SEAP: Secure and efficient authentication protocol for NFC applications using pseudonyms. *IEEE Trans. Consum. Electron.* **2016**, *62*, 30–38. [CrossRef]
12. Chatterjee, S.; Das, A.; Sing, J. An Enhanced Access Control Scheme in Wireless Sensor Networks. *Ad-Hoc Sens. Wirel. Netw.* **2014**, *21*, 121–149.
13. Mishra, D.; Das, A.K.; Mukhopadhyay, S. A secure and efficient ECC-based user anonymity-preserving session initiation authentication protocol using smart card. *Peer- Netw. Appl.* **2016**, *9*, 171–192. [CrossRef]
14. Challa, S.; Das, A.K.; Gope, P.; Kumar, N.; Wu, F.; Vasilakos, A.V. Design and analysis of authenticated key agreement scheme in cloud-assisted cyber–physical systems. *Future Gener. Comput. Syst.* **2020**, *108*, 1267–1286. [CrossRef]
15. Das, A.K.; Sutrala, A.K.; Kumari, S.; Odelu, V.; Wazid, M.; Li, X. An efficient multi-gateway-based three-factor user authentication and key agreement scheme in hierarchical wireless sensor networks. *Secur. Commun. Netw.* **2016**, *9*, 2070–2092. [CrossRef]
16. Lin, C.; He, D.; Kumar, N.; Choo, K.R.; Vinel, A.; Huang, X. Security and Privacy for the Internet of Drones: Challenges and Solutions. *IEEE Commun. Mag.* **2018**, *56*, 64–69. [CrossRef]
17. Wazid, M.; Das, A.K.; Khan, M.K.; Al-Ghaiheb, A.A.; Kumar, N.; Vasilakos, A.V. Secure Authentication Scheme for Medicine Anti-Counterfeiting System in IoT Environment. *IEEE Internet Things J.* **2017**, *4*, 1634–1646. [CrossRef]
18. Wazid, M.; Bagga, P.; Das, A.K.; Shetty, S.; Rodrigues, J.J.P.C.; Park, Y. AKM-IoV: Authenticated Key Management Protocol in Fog Computing-Based Internet of Vehicles Deployment. *IEEE Internet Things J.* **2019**, *6*, 8804–8817. [CrossRef]
19. Srinivas, J.; Das, A.K.; Wazid, M.; Kumar, N. Anonymous Lightweight Chaotic Map-Based Authenticated Key Agreement Protocol for Industrial Internet of Things. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 1133–1146. [CrossRef]
20. Bera, B.; Das, A.K.; Sutrala, A.K. Private blockchain-based access control mechanism for unauthorized UAV detection and mitigation in Internet of Drones environment. *Comput. Commun.* **2021**, *166*, 91–109. [CrossRef]

21. Li, C.; Lee, C.; Weng, C. Security and Efficiency Enhancement of Robust ID Based Mutual Authentication and Key Agreement Scheme Preserving User Anonymity in Mobile Networks. *J. Inf. Sci. Eng.* **2018**, *34*, 155–170.
22. Bera, B.; Saha, S.; Das, A.K.; Kumar, N.; Lorenz, P.; Alazab, M. Blockchain-Envisioned Secure Data Delivery and Collection Scheme for 5G-Based IoT-Enabled Internet of Drones Environment. *IEEE Trans. Veh. Technol.* **2020**. [CrossRef]
23. Srinivas, J.; Das, A.K.; Kumar, N.; Rodrigues, J.J.P.C. TCALAS: Temporal Credential-Based Anonymous Lightweight Authentication Scheme for Internet of Drones Environment. *IEEE Trans. Veh. Technol.* **2019**, *68*, 6903–6916. [CrossRef]
24. Jiang, Q.; Zhang, N.; Ni, J.; Ma, J.; Ma, X.; Choo, K.K.R. Unified Biometric Privacy Preserving Three-Factor Authentication and Key Agreement for Cloud-Assisted Autonomous Vehicles. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9390–9401. [CrossRef]
25. Wazid, M.; Das, A.K.; Lee, J.H. Authentication protocols for the internet of drones: Taxonomy, analysis and future directions. *J. Ambient. Intell. Humaniz. Comput.* **2018**. [CrossRef]
26. Li, C.T.; Chen, C.L.; Lee, C.C.; Weng, C.Y.; Chen, C.M. A novel three-party password-based authenticated key exchange protocol with user anonymity based on chaotic maps. *Soft Comput.* **2018**, *22*, 2495–2506. [CrossRef]
27. Wazid, M.; Bera, B.; Mitra, A.; Das, A.K.; Ali, R. Private Blockchain-Envisioned Security Framework for AI-Enabled IoT-Based Drone-Aided Healthcare Services. In Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond (DroneCom'20), London, UK, 25 September 2020; pp. 37–42.
28. Bera, B.; Das, A.K.; Garg, S.; Piran, M.J.; Hossain, M.S. Access Control Protocol for Battlefield Surveillance in Drone-Assisted IoT Environment. *IEEE Internet Things J.* **2021**. [CrossRef]
29. Zhang, Y.; He, D.; Li, L.; Chen, B. A lightweight authentication and key agreement scheme for Internet of Drones. *Comput. Commun.* **2020**, *154*, 455–464. [CrossRef]
30. Gope, P. LAAP: Lightweight anonymous authentication protocol for D2D-Aided fog computing paradigm. *Comput. Secur.* **2019**, *86*, 223–237. [CrossRef]
31. Canetti, R.; Krawczyk, H. Analysis of key-exchange protocols and their use for building secure channels. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Innsbruck, Austria, 6–10 May 2001; pp. 453–474.
32. Concone, F.; Re, G.L.; Morana, M. SMCP: A Secure Mobile Crowdsensing Protocol for fog-based applications. *Hum.-Centric Comput. Inf. Sci.* **2020**, *10*, 1–23. [CrossRef]
33. Basudan, S.; Lin, X.; Sankaranarayanan, K. A Privacy-Preserving Vehicular Crowdsensing-Based Road Surface Condition Monitoring System Using Fog Computing. *IEEE Internet Things J.* **2017**, *4*, 772–782. [CrossRef]
34. Cui, M.; Han, D.; Wang, J. An efficient and safe road condition monitoring authentication scheme based on fog computing. *IEEE Internet Things J.* **2019**, *6*, 9076–9084. [CrossRef]
35. Guo, Y.; Zhang, Z.; Guo, Y. Fog-Centric Authenticated Key Agreement Scheme Without Trusted Parties. *IEEE Syst. J.* **2020**, *2020*, 1–10. [CrossRef]
36. Ali, Z.; Chaudhry, S.A.; Mahmood, K.; Garg, S.; Lv, Z.; Zikria, Y.B. A clogging resistant secure authentication scheme for fog computing services. *Comput. Netw.* **2020**, *185*, 107731. [CrossRef]
37. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
38. Messerges, T.S.; Dabbish, E.A.; Sloan, R.H. Examining smart-card security under the threat of power analysis attacks. *IEEE Trans. Comput.* **2002**, *51*, 541–552. [CrossRef]
39. May, W.E. Secure Hash Standard, 2015. Available online: http://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf (accessed on 10 February 2021).
40. Abdalla, M.; Fouque, P.; Pointcheval, D. Password-based authenticated key exchange in the three-party setting. In Proceedings of the 8th International Workshop on Theory and Practice in Public Key Cryptography (PKC'05), Lecture Notes in Computer Science (LNCS), Les Diablerets, Switzerland, 23–26 January 2005; pp. 65–84.
41. AVISPA. Automated Validation of Internet Security Protocols and Applications. 2021. Available online: http://www.avispa-project.org/ (accessed on 10 January 2021).
42. Wazid, M.; Das, A.K.; Odelu, V.; Kumar, N.; Susilo, W. Secure Remote User Authenticated Key Establishment Protocol for Smart Home Environment. *IEEE Trans. Dependable Secur. Comput.* **2020**, *17*, 391–406. [CrossRef]
43. Chang, C.C.; Le, H.D. A provably secure, efficient, and flexible authentication scheme for ad hoc wireless sensor networks. *IEEE Trans. Wirel. Commun.* **2016**, *15*, 357–366. [CrossRef]
44. Sarkar, P. A Simple and Generic Construction of Authenticated Encryption with Associated Data. *ACM Trans. Inf. Syst. Secur.* **2010**, *13*, 33. [CrossRef]
45. Banerjee, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Rodrigues, J.J.; Park, Y. Physically secure lightweight anonymous user authentication protocol for internet of things using physically unclonable functions. *IEEE Access* **2019**, *7*, 85627–85644. [CrossRef]
46. Banerjee, S.; Roy, S.; Odelu, V.; Das, A.K.; Chattopadhyay, S.; Rodrigues, J.J.; Park, Y. Multi-Authority CP-ABE-Based user access control scheme with constant-size key and ciphertext for IoT deployment. *J. Inf. Secur. Appl.* **2020**, *53*, 102503. [CrossRef]
47. von Oheimb, D. The high-level protocol specification language hlpsl developed in the eu project avispa. In Proceedings of the 3rd APPSEM II (Applied Semantics II) Workshop (APPSEM'05), Frauenchiemsee, Germany, 12–15 September 2005; pp. 1–17.
48. AVISPA. SPAN, the Security Protocol ANimator for AVISPA. 2021. Available online: http://www.avispa-project.org/ (accessed on 10 January 2021).

49. MIRACL Cryptographic SDK: Multiprecision Integer and Rational Arithmetic Cryptographic Library. 2020. Available online: https://github.com/miracl/MIRACL (accessed on 10 March 2021).
50. Advanced Encryption Standard (AES), 2001. FIPS PUB 197, National Institute of Standards and Technology (NIST), U.S. Department of Commerce, November 2001. Available online: http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf (accessed on 10 February 2021).
51. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy Extractors: How to Generate Strong Keys from Biometrics and Other Noisy Data. *SIAM J. Comput.* **2008**, *38*, 97–139. [CrossRef]
52. Knuth, D.E. *The Art of Computer Programming: Seminumerical Algorithms*, 3rd ed.; Addison-Wesley Longman Publishing Co., Inc.: Boston, MA, USA, 1997; Volume 2.
53. He, D.; Zeadally, S.; Xu, B.; Huang, X. An Efficient Identity-Based Conditional Privacy-Preserving Authentication Scheme for Vehicular Ad Hoc Networks. *IEEE Trans. Inf. Forensics Secur.* **2015**, *10*, 2681–2691. [CrossRef]
54. Raspberry Pi 3 Model B+. 2020. Available online: https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/ (accessed on 10 May 2021).